Unmarked version of all claims:

1. (Currently amended) A computer-implemented method for accelerating database query processing, comprising:

determining during execution of a particular query whether continued execution of a particular query execution plan is worthwhile by

calculating the amount of query execution remaining;

computing the difference between estimated optimization parameter values and actual optimization parameter values to determine the significance of parameter estimation errors;

concluding that continued execution is not worthwhile if a significant amount of query execution remains and significant parameter estimation errors have occurred; and

if continued execution is not worthwhile, then suspending query execution, re-optimizing the query, and restarting query execution with a re-optimized query plan.

2. (Currently amended) The method of claim 1 wherein the re-optimizing further comprises:

generating a number of alternative query execution plans, including plans generated with and plans generated without using temporary results computed in prior executions;

assigning a cost to each alternative plan that reflects plan optimality; and

choosing the optimal alternative as the re-optimized query plan.

3. (Original) The method of claim 2 further comprising exploiting actual optimization parameter values during the re-optimizing.

4. (Original) The method of claim 3 wherein the actual optimization parameters include at least one of: cardinality, memory, communication costs, and I/O operations.

5. (Original) The method of claim 1 further comprising selectively reusing materialized partial query results during subsequent re-optimizations and query executions, if the reuse reduces overall computational costs.

6. (Currently amended) The method of claim 1 further comprising selectively retaining temporarily materialized views storing partial query results until a lazy removal condition occurs, said conditions including (a) updating of at least one table contributing to a materialized view, and (b) determining that new storage space is needed for other materialized views.

7. (Original) The method of claim 1 further comprising selectively returning records at each execution cycle if the records have not previously been returned.

8. (Currently amended) The method of claim 1 wherein records returned during previous executions are eliminated from answer sets returned in subsequent executions.

9. (Original) The method of claim 8 wherein the returned records are identified by a unique derived record ID assigned to records during query execution.

10. (Original) The method of claim 1 further comprising placing a number of checkpoints in the query execution plan to compute the difference between estimated optimization parameter values and actual optimization parameter values.

11. (Original) The method of claim 10 wherein the checkpoints are placed at points in the query execution plan where an entire intermediate result is materialized before proceeding with further operators in the plan.

12. (Original) The method of claim 11 wherein an explicit materialization is added to the query execution plan, just before the checkpoint, to materialize the intermediate result.

13. (Original) The method of claim 10 wherein the checkpoint is pushed below a materialization point for subsequent execution.

14. (Original) The method of claim 13 further comprising buffering rows until the checkpoint is evaluated, enabling pipelining with some delay.

15. (Original) The method of claim 14 wherein exhaustion of temporary space triggers a re-optimization instead of signaling an error.

16. (Original) The method of claim 13 further comprising transferring each row to its parent operator in a pipelined manner, storing identifiers of all rows returned on a side table using an INSERT plan operator just below the return operator, then compensating for returned row results by executing an anti join between the side table and a new result stream.

17. (Currently amended) A computer-implemented system for accelerating database query processing, comprising:

means for determining during execution of a particular query whether continued execution of a particular query execution plan is worthwhile by

calculating the amount of query execution remaining;

computing the difference between estimated optimization parameter values and actual optimization parameter values; and

concluding that continued execution is not worthwhile if a significant amount of query execution remains and significant parameter estimation errors have occurred; and

means for, if continued execution is not worthwhile, then performing the steps of suspending query execution, re-optimizing the query, and restarting query execution with a re-optimized query plan.

18. (Currently amended) A computer program product tangibly embodying a program of computer-executable instructions to perform a method for accelerating database query processing, the method comprising:

determining during execution of a particular query whether continued execution of a particular query execution plan is worthwhile by

calculating the amount of query execution remaining;

computing the difference between estimated optimization parameter values and actual optimization parameter values; and

concluding that continued execution is not worthwhile if a significant amount of query execution remains and significant parameter estimation errors have occurred; and

if continued execution is not worthwhile, then suspending query execution, re-optimizing the query, and restarting query execution with a re-optimized query plan.